Студент группы ИС-23

Ивахненко А.Н

Практическое занятие No 6

Тема: составление программ с функциями в IDE PyCharm Community.

Цель практического занятия : закрепить усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ со списками в IDE PyCharm Community.

Постановка задачи №1:

1. Дан целочисленный список размера 10. Вывести вначале все содержащиеся в данном списке четные числа в порядке возрастания их индексов, а затем — все нечетные числа в порядке убывания их индексов.

Код:

```
my_list = [23, 14, 8, 17, 5, 10, 32, 20, 11, 7]

# Инициализация списков для четных и нечетных чисел
even_numbers = []
odd_numbers = []

# Перебор элементов списка и добавление их в соответствующие списки
for i in range(len(my_list)):
    if my_list[i] % 2 == 0:
        even_numbers.append(my_list[i])
    else:
        odd_numbers.append(my_list[i])

# Вывод результатов
print("Четные числа в порядке возрастания индексов:", even_numbers)
print("Нечетные числа в порядке убывания индексов:", odd_numbers[::-1])
```

Протокол выполнения программы:

1. Создается исходный список my_list
2. Инициализируются пустые списки even_numbers и odd_numbers
3. Программа начинает перебор элементов в my_list поочередно
4. Каждое число проверяется на четность.
5. Когда все числа в my_list обработаны, программа завершает цикл
6. ***Вывод результата:***
   Четные числа в порядке возрастания индексов: [14, 8, 10, 32, 20]
   Нечетные числа в порядке убывания индексов: [7, 11, 5, 17, 23]

Постановка задачи №2:

2. Дан список размера N. Найти количество участков, на которых его элементы монотонно убывают.

Код:

```
1. def count_decreasing_sections(lst):
2.     decreasing_sections = 0
3.     length = len(lst)
4.
5.     # Проверка каждой пары соседних элементов в списке
6.     for i in range(1, length):
7.         # Если текущий элемент меньше предыдущего, это начало нового
   участка убывания
8.         if lst[i] < lst[i - 1]:
9.             decreasing_sections += 1
10.
11.    return decreasing_sections
12.
13.# Пример списка для тестирования
14.my_list = [5, 4, 3, 8, 6, 2, 1, 7, 9]
15.
16.# Получение количества участков, на которых элементы убывают
17.result = count_decreasing_sections(my_list)
18.print("Количество участков убывания:", result)
```

Протокол выполнения программы:

1. Определена функция count_decreasing_sections(lst), которая считает количество участков убывания в списке.

2. Для этого используется генератор списка, который подсчитывает количество случаев, когда следующий элемент меньше предыдущего.

3. Подсчитанное количество участков убывания выводится на экран для списка [5, 4, 3, 8, 6, 2, 1, 7, 9].

4. ***Вывод результата***: Количество участков убывания: 5

Постановка задачи №3:

Дано множество A из N точек на плоскости и точка B (точки заданы своими координатами x, y). Найти точку из множества A, наиболее близкую к точке B. Расстояние R между точками с координатами (x1, y1) и (x2, y2) вычисляется по формуле: $R = \sqrt{(x2 – x1)2 + (y2 – y1)2}$. Для хранения данных о каждом наборе точек следует использовать по два список: первый список для хранения абсцисс, второй — для хранения ординат.

Код:

```python
def distance(point1, point2):
    return ((point2[0] - point1[0]) ** 2 + (point2[1] - point1[1]) ** 2) ** 0.5

def closest_point_to_b(points_x, points_y, point_b):
    min_distance = float('inf')  # Инициализация минимального расстояния
    closest_point = None  # Инициализация ближайшей точки

    for i in range(len(points_x)):
        # Вычисление расстояния между точкой из множества A и точкой B
        dist = distance((points_x[i], points_y[i]), point_b)

        # Если текущее расстояние меньше минимального, обновляем значения
        if dist < min_distance:
            min_distance = dist
            closest_point = (points_x[i], points_y[i])

    return closest_point

# Пример данных
points_x = [1, 3, 7, 9, 12]
points_y = [4, 6, 2, 15, 9]
point_b = (8, 10)

# Поиск ближайшей точки к точке B из множества A
closest = closest_point_to_b(points_x, points_y, point_b)
print("Ближайшая точка к B из множества A:", closest)
```

Протокол выполнения программы:

1.Функция distance(point1, point2) вычисляет расстояние между двумя точками на плоскости по формуле Евклида.

2.Функция closest_point_to_b(points_x, points_y, point_b) находит ближайшую точку из списка координат points_x и points_y к точке point_b.

3.Созданы списки points_x и points_y с координатами точек из множества A, а также задана точка point_b.

4.Вызвана функция closest_point_to_b(points_x, points_y, point_b) для поиска ближайшей точки к точке point_b из множества A.

5.Вывод результата:

Ближайшая точка к B из множества A: (12, 9)

Вывод: Я закрепил усвоенные знания, понятия, алгоритмы, основные принципы составления программ, приобрести навыки составление программ со списками в IDE PyCharm Community.